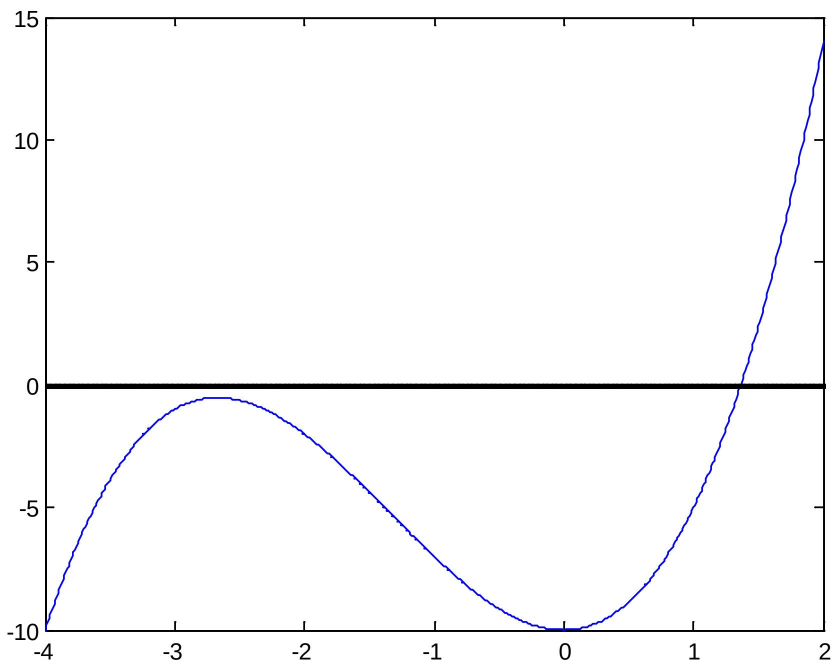
ALGORÍTMICA NUMÉRICA **12 Enero 2012**

Examen Computacional 3: Solucion

% Ejercicio 1.1

```
function fx=fun7(x)
fx=x.^3+4*x.^2-10;
return

x=-4:0.01:2;plot(x,fun7(x),x,0,'k-')
```



Como se puede comprobar en la figura, f(x) tiene una raiz en el intervalo [a,b]=[1,2].

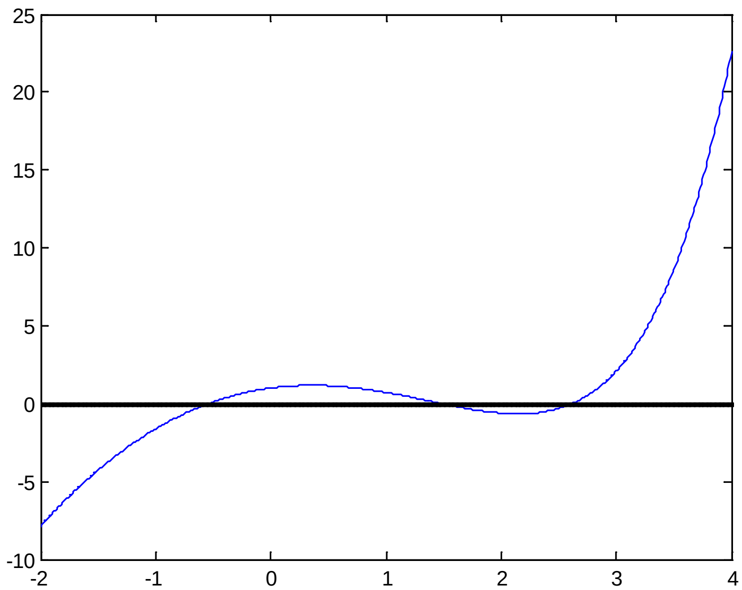
```
x=1
for k=1:10
    x=(2*x^3+4*x^2+10)/(3*x^2+8*x);
    fprintf('Iteración: %d valor x_k: %.16f valor f(x_k): %.3e\n',k,x,fun7(x));   %
Vuelco los valores k, x
end
s=x
 x =
     1
Iteración: 1 valor x_k: 1.4545454545454546 valor f(x_k): 1.540e+000
Iteración: 2 valor x_k: 1.3689004010695187 valor f(x_k): 6.072e-002
Iteración: 3 valor x_k: 1.3652366002021157 valor f(x_k): 1.088e-004
Iteración: 4 valor x_k: 1.3652300134353668 valor f(x_k): 3.512e-010
Iteración: 5 valor x_k: 1.3652300134140969 valor f(x_k): 0.000e+000
Iteración: 6 valor x_k: 1.3652300134140967 valor f(x_k): -3.553e-015
Iteración: 7 valor x_k: 1.3652300134140971 valor f(x_k): 3.553e-015
Iteración: 8 valor x_k: 1.3652300134140969 valor f(x_k): 0.000e+000
Iteración: 9 valor x_k: 1.3652300134140967 valor f(x_k): -3.553e-015
Iteración: 10 valor x_k: 1.3652300134140971 valor f(x_k): 3.553e-015
```

```
s =

    1.3652
El valor de la raíz s=1.36523000134.

x=1
for k=1:5
    x=(2*x^3+4*x^2+10)/(3*x^2+8*x);Erel=abs(x-s)/abs(s);
    fprintf('Iteración: %d valor x_k: %.16f valor f(x_k): %.3e num. cifras: 
%3d\n',k,x,fun7(x),floor(-log10(Erel)));   % Vuelco los valores k, x
end

Iteración: 1 valor x_k: 1.4545454545454546 valor f(x_k): 1.540e+000 num. cifras: 
1
Iteración: 2 valor x_k: 1.3689004010695187 valor f(x_k): 6.072e-002 num. cifras: 
2
Iteración: 3 valor x_k: 1.3652366002021157 valor f(x_k): 1.088e-004 num. cifras: 
5
Iteración: 4 valor x_k: 1.3652300134353668 valor f(x_k): 3.512e-010 num. cifras: 
10
Iteración: 5 valor x_k: 1.3652300134140969 valor f(x_k): 0.000e+000 num. cifras: 
15
```

En cada iteración se duplica el número de cifras significativas, luego la velocidad de convergencia es cuadrática.

% Ejercicio 1.2

```
function fx=fun8(x)
fx=exp(x)-2*x.^2;
return

x=-2:0.01:5;plot(x,fun8(x),x,0,'k-')
```




```
x=0;
```

```
for k=1:5
    x=(exp(x)*(x-1)-2*x^2)/(exp(x)-4*x);
    fprintf('Iteración: %d valor x_k: %.16f valor g(x_k): %.3e\n',k,x,fun8(x));   %
Vuelco los valores k, x
end
s1=x
x=1;
for k=1:5
    x=(exp(x)*(x-1)-2*x^2)/(exp(x)-4*x);
    fprintf('Iteración: %d valor x_k: %.16f valor g(x_k): %.3e\n',k,x,fun8(x));   %
Vuelco los valores k, x
end
s2=x
x=3;
for k=1:5
    x=(exp(x)*(x-1)-2*x^2)/(exp(x)-4*x);
    fprintf('Iteración: %d valor x_k: %.16f valor g(x_k): %.3e\n',k,x,fun8(x));   %
Vuelco los valores k, x
end
s3=x
```

```
Iteración: 1 valor x_k: -1.0000000000000000 valor g(x_k): -1.632e+000
Iteración: 2 valor x_k: -0.6263357126013460 valor g(x_k): -2.500e-001
Iteración: 3 valor x_k: -0.5440807902887221 valor g(x_k): -1.167e-002
Iteración: 4 valor x_k: -0.5398464536457096 valor g(x_k): -3.065e-005
Iteración: 5 valor x_k: -0.5398352769806530 valor g(x_k): -2.134e-010

s1 =

  -0.539835276980653

Iteración: 1 valor x_k: 1.5604054342113962 valor g(x_k): -1.090e-001
Iteración: 2 valor x_k: 1.4868141259830758 valor g(x_k): 1.749e-003
Iteración: 3 valor x_k: 1.4879618819282945 valor g(x_k): 2.797e-007
Iteración: 4 valor x_k: 1.4879620654981722 valor g(x_k): 7.105e-015
Iteración: 5 valor x_k: 1.4879620654981773 valor g(x_k): 0.000e+000

s2 =

   1.487962065498177

Iteración: 1 valor x_k: 2.7420657473956029 valor g(x_k): 4.812e-001
Iteración: 2 valor x_k: 2.6363333925397026 valor g(x_k): 6.141e-002
Iteración: 3 valor x_k: 2.6183595116862346 valor g(x_k): 1.596e-003
Iteración: 4 valor x_k: 2.6178669771770728 valor g(x_k): 1.178e-006
Iteración: 5 valor x_k: 2.6178666130670116 valor g(x_k): 6.430e-013

s3 =

   2.617866613067012
```

Los valores mostrados son las raices s1, s2 y s3.

```
>> x=3;
for k=1:5
    x=(exp(x)*(x-1)-2*x^2)/(exp(x)-4*x);Erel=abs(x-s3)/abs(s3);
    fprintf('Iteración: %d valor x_k: %.16f valor g(x_k): %.3e num. cifras:
%3d\n',k,x,fun8(x),floor(-log10(Erel)));
end
```

```
Iteración: 1 valor x_k: 2.7420657473956029 valor g(x_k): 4.812e-001 num. cifras:
1
Iteración: 2 valor x_k: 2.6363333925397026 valor g(x_k): 6.141e-002 num. cifras:
2
Iteración: 3 valor x_k: 2.6183595116862346 valor g(x_k): 1.596e-003 num. cifras:
3
Iteración: 4 valor x_k: 2.6178669771770728 valor g(x_k): 1.178e-006 num. cifras:
6
Iteración: 5 valor x_k: 2.6178666130670116 valor g(x_k): 6.430e-013 num. cifras:
Inf
```

Con 5 iteraciones alcanzamos la precisión de la máquina. Igual que en el apartado anterior, la convergencia es cuadrática.
En x=0, g'(0) es positivo y la recta tangente a g(x) en 0 corta al eje en un punto situado a la izquierda de x=0 (valores negativos), y la raíz aproximada es negativa. En x=1, g'(1) es negativa y la recta recta tangente a g(x) en 1 corta al eje en un punto situado a la derecha de x=1, y la raíz aproximada es s2>1. (raíz positiva).
A partir de la gráfica de g(x), el máximo lo alcanza en aprox. 0.5 y el mínimo en 2.2, luego el intervalo de convergencia de Newton de la raíz s2 es aprox [0.5, 2.2].

% Ejercicio 2

```
>> N=6;J=ones(N);H=3*eye(N)-triu(J,-1)+triu(J,2)
[L,U]=lu(H)
L*U-H

H =

     2    -1     0     0     0     0
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
     0     0     0    -1     2    -1
     0     0     0     0    -1     2


L =

    1.0000         0         0         0         0         0
   -0.5000    1.0000         0         0         0         0
         0   -0.6667    1.0000         0         0         0
         0         0   -0.7500    1.0000         0         0
         0         0         0   -0.8000    1.0000         0
         0         0         0         0   -0.8333    1.0000


U =

    2.0000   -1.0000         0         0         0         0
         0    1.5000   -1.0000         0         0         0
         0         0    1.3333   -1.0000         0         0
         0         0         0    1.2500   -1.0000         0
         0         0         0         0    1.2000   -1.0000
         0         0         0         0         0    1.1667


ans =
```

```
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0

function X=inversa(A)
     [L,U]=lu(A);
     N=length(A);
     b=eye(N);X=zeros(N);y=zeros(N);
     for k=1:N
         y(:,k)=solve_L(L,b(:,k));
         X(:,k)=solve_U(U,y(:,k));
     end
return

>> X=inversa2(H)

X =

    0.8571    0.7143    0.5714    0.4286    0.2857    0.1429
    0.7143    1.4286    1.1429    0.8571    0.5714    0.2857
    0.5714    1.1429    1.7143    1.2857    0.8571    0.4286
    0.4286    0.8571    1.2857    1.7143    1.1429    0.5714
    0.2857    0.5714    0.8571    1.1429    1.4286    0.7143
    0.1429    0.2857    0.4286    0.5714    0.7143    0.8571

>> X*H-eye(6)

ans =

  1.0e-015 *

   -0.1110   -0.1110   -0.1110    0.1665    0.0278         0
         0   -0.2220   -0.2220    0.3331    0.0555         0
         0         0   -0.2220    0.4441         0         0
         0         0   -0.4441    0.6661   -0.1110         0
         0         0   -0.2220         0    0.2220         0
         0         0   -0.1110    0.2220         0         0

>> precision=norm(X*H-eye(length(H)))

precision =

  1.0829e-015



>>  N=500;J=ones(N);H=3*eye(N)-triu(J,-1)+triu(J,2);

     tic
     X=inversa(H);
     toc
     precision_inversa=norm(X*H-eye(length(H)))
     tic
     X=inv(H);
     toc
     precision_inv=norm(X*H-eye(length(H)))
```

```
Elapsed time is 7.982828 seconds.

precision_inversa =

  3.4912e-011

Elapsed time is 0.180253 seconds.

precision_inv =

  8.6747e-012
```

La function inv es mas precisa numéricamente que la función inversa, ya precision_inv = 8.6747e-012 es menor que precision_inversa = 3.4912e-011.

Como 3.4912e-011/8.6747e-012 = 4.0246, la función inv es 4 veces mas precisa que la función inversa.

Por otro lado, la función inv es mas rápida que la función inversa, ya que el tiempo de ejecución de la primera ha sido 0.180253 seconds mientras que el tiempo de ejecución de la segunda ha sido de 7.982828 seconds. Como 7.982828/0.180253= 44.2868, la función inv es 44 veces mas rápida que la función inversa.